Н.Т. Абдуллаев, К.Ш. Пашаева

Использование модели машинного обучения для классификации миографических заболеваний

Аннотация

Для решения задачи классификации применены модели последовательного машинного обучения с использованием методов глубокого обучения. Работа реализована на платформе Kaggle с использованием языка программирования Python. Установлена зависящая от числа эпох точность нейронной сети при помощи кривых точности и потерь при обучении, оценивается влияние числа потерь на повышение точности сети.

Введение

Электромиография – метод исследования и оценки функционального состояния периферических нервов, спинно-мозговых корешков и мышц. В основе его лежит оценка электрической проводимости и возбудимости нервных структур и мышц. Современные методы обработки применяются практически во всех областях медицины, включая электромиографию, а методы искусственного интеллекта успешно используются для повышения результатов диагностики.

В [1] исследователи предлагают метод классификации трех типов сигналов мембранного потенциала покоя, полученных в виде изображений посредством диагностической игольчатой электромиографии (ЭМГ) с использованием TensorFlow-Slim и Python для реализации схемы распознавания изображений на основе искусственного интеллекта.

Классификационная модель Inception v4 использовалась для разделения изображений сигналов на три категории (точность = 93,8 %, точность = 99,5 %, полнота = 90,8 %). Это было сделано путем применения предварительно обученной модели Inception v4 к методу тонкой настройки. Модель распознавания изображений была создана для обучения с использованием различных типов медицинских данных на основе изображений.

Авторы статьи [2] представляют электромиографический (ЭМГ) метод распознавания образов с целью определения команд движения для управления протезом руки путем накопления доказательств на основе искусственного интеллекта с несколькими параметрами. Интегральное абсолютное значение, дисперсия, коэффициенты авторегрессионной (AR) модели, коэффициенты линейного кепстра и вектор адаптивного кепстра извлекаются как параметры признаков из нескольких временных сегментов сигналов ЭМГ. Распознавание образов осуществляется посредством процедуры накопления свидетельств с использованием расстояний, измеренных с эталонными параметрами. Функция нечеткого отображения предназначена для преобразования расстояний с целью применения метода накопления доказательств. Представлены результаты, подтверждающие осуществимость предложенного подхода к распознаванию образов ЭМГ.

В [3] для классификации миоэлектрических сигналов выбраны шесть уникальных движений рук. Характеристики, выбранные для сигнала ЭМГ, относятся к временной и частотно-временной областям. В данной работе демонстрируются возможности системы распознавания образов ЭМГ, использующей ANFIS в качестве классификатора с методом обучения в реальном времени. Однако наши результаты показывают, что используемый подход ANFIS в реальном времени вместе с пользовательской оценкой обеспечивает среднюю точность 96,7 %. Эта скорость превосходит ранее опубликованный результат с использованием метода 1 в реальном времени искусственных нейронных сетей (ИНС).

В работе [4] представлен новый многодоменный алгоритм обучения под названием ADANN (Adaptive Domain Adversarial Neural Network), который значительно повышает (p = 0,00004) точность межпредметной классификации, в среднем на 19,40 % по сравнению со стандартным обучением. Используя функции, сгенерированные ADANN, работа [4] обеспечивает первый топологический анализ данных распознавания жестов на основе ЭМГ для характеристики информации, закодированной в глубокой сети, с применением созданных вручную функций в качестве ориентиров. Этот анализ показывает, что созданные вручную функции и изученные функции (на более ранних уровнях) пытаются различать все жесты, но для этого не кодируется одна и та же информация. На более поздних уровнях изученные функции склонны вместо этого принимать стратегию «один против всех» для данного класса. Кроме того, при помощи методов визуализации сверточной сети было выявлено, что изученные функции фактически имеют тенденцию игнорировать наиболее активный канал во время сокращения, что резко контрастирует с преобладанием созданных вручную функций, предназначенных для сбора информации об амплитуде.

Целью статьи [5] является исследование подхода глубокой нейронной сети к классификации 41-го движения руки и запястья на основе сигнала sEMG. Предложенные модели были обучены и оценены с использованием общедоступной базы данных проекта Ninapro, одной из крупнейших общедоступных баз данных sEMG для усовершенствованного миоэлектрического протезирования руки. Для этого исследования использовали два набора данных: DB5 с недорогими 16 каналами и настройкой частоты дискретизации 200 Гц и DB7 с 12 каналами и настройкой частоты дискретизации 2 кГц. Подход авторов достиг общей точности 93,87 ± 1,49 и 91,69 ± 4,68 % со сбалансированной точностью 84,00 ± 3,40 и 84,66 ± 4,78 % для DB5 и DB7 соответственно.

Материалы и методы

Машинное обучение (англ. machine learning, ML) – класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счет применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

Последовательная модель – это упрощенная версия функциональной модели, простейшая линейная сквозная структурная последовательность без бифуркации и линейный стек из нескольких сетевых уровней.

Keras реализует множество уровней, включая базовый уровень ядра, уровень свертки Convolution, уровень пула и другие очень богатые и интересные сетевые структуры.

Библиотека Keras Python позволяет быстро и легко разрабатывать модели глубокого обучения. Последовательный API позволяет создавать модели «слой за слоем» для большинства задач. Он ограничен тем, что не позволяет создавать модели с общими слоями или с несколькими входами и выходами.

Функциональный API в Keras – это альтернативный способ создания моделей, который предлагает гораздо большую гибкость, включая создание более сложных моделей [6].

LabelBinarizer – это служебный класс, помогающий создать матрицу индикаторов меток из списка мультиклассовых ме-

ток. Использование этого формата может включать в себя мультиклассовую классификацию в средствах оценки, которые поддерживают формат матрицы индикатора метки.

LabelBinarizer не нужен, если вы используете средство оценки, которое уже поддерживает мультиклассовые данные. В мультиметочном обучении совместный набор задач бинарной классификации выражается массивом бинарных индикаторов меток: каждая выборка представляет собой одну строку двумерного массива формы (n_samples, n_classes) с двоичными значениями, где единица, т. е. ненулевые элементы, соответствует подмножеству меток для этого образца. Массив, такой как np.array([[1, 0, 0], [0, 1, 1], [0, 0, 0]]) представляет метку 0 в первом примере, метки 1 и 2 во втором примере, а в третьем примере меток нет.

LabelEncoder – это служебный класс, помогающий нормализовать метки таким образом, чтобы они содержали только значения от 0 до n_classes-1. Иногда это полезно для написания эффективных подпрограмм Cython. LabelEncoder можно использовать следующим образом: для преобразования нечисловых меток (при условии, что они поддаются хэшированию и сопоставимы) в числовые метки [7].

Последовательная модель подходит для простого стека слоев, где каждый слой имеет ровно один входной тензор и один выходной тензор. Последовательная модель не подходит, когда:

- модель имеет несколько входов или несколько выходов;
- любой из слоев имеет несколько входов или несколько выходов;
- нужно сделать общий доступ к слою;
- нужна нелинейная топология (например, остаточное соединение, многоветвевая модель).

Можно создать модель Sequential, передав список слоев конструкторам Sequential. Слои доступны через атрибут Layers. Также можно создать модель Sequential постепенно, при помощи метода add():; существует также соответствующий метод pop() для удаления слоев: модель Sequential ведет себя очень похоже на список слоев. Также конструктор Sequential принимает аргумент имени, как и любой слой или модель в Keras. Это полезно для аннотирования графов TensorBoard семантически значимыми именами.

Как правило, все слои в Keras должны знать форму своих входных данных, чтобы иметь возможность создавать свои веса. Итак, когда создается такой слой, изначально он не имеет весов. Он создает свои веса при первом обращении к входным данным, поскольку форма весов зависит от формы входных данных. Естественно, это относится и к моделям Sequential. Когда модель Sequential создается без входной формы, она «не построена»: у нее нет весов (и вызов model.weights приводит к ошибке, указывающей именно это). Веса создаются, когда модель впервые видит некоторые входные данные. Однако при поэтапном построении модели Sequential может быть очень полезно иметь возможность отображать сводку модели на данный момент, включая текущую выходную форму. В этом случае нужно запустить свою модель, передав входной объект вашей модели, чтобы она знала свою входную форму с самого начала. Модели, построенные с предопределенной входной формой, подобной этой, всегда имеют вес (даже до того, как будут видны какие-либо данные) и всегда имеют определенную выходную форму [8].

При построении новой последовательной архитектуры полезно постепенно складывать слои при помощи add() и часто печатать сводки модели. Например, это позволяет отслеживать, как стек слоев Conv2D и MaxPooling2D снижает дискретизацию карт объектов изображения.

После создания последовательной модели она ведет себя как функциональная модель API. Это означает, что каждый слой имеет входной и выходной атрибуты. Эти атрибуты можно использовать для выполнения изящных действий, таких как быстрое создание модели, которая извлекает выходные данные всех промежуточных слоев в последовательной модели.

Трансферное обучение состоит из замораживания нижних слоев модели и обучения только верхних слоев, распространения плана трансферного обучения с использованием последовательных моделей.

Модель предоставляет функции для обучения, оценки и процесса прогнозирования. Они следующие [9]-[12]:

- компилировать – настроить процесс обучения модели;
- fit – обучить модель, используя обучающие данные;
- оценить – оценить модель, используя тестовые данные;
- предсказать – предсказать результаты для нового ввода.

Объектно-ориентированное программирование (ООП) – наиболее эффективная модель, действующая путем создания подкласса класса Model: в этом случае нужно определить свои слои в init и реализовать прямой проход модели при вызове.

Эта работа выполняется на платформе Kaggle с использованием языка программирования Python. Модель, которая используется для получения прогноза, основана на модели последовательного машинного обучения с применением методов глубокого обучения [13].

Для эксперимента выбраны электромиографические сигналы здорового пациента и три часто встречающиеся болезни, такие как карпальный туннельный синдром, кубитальный туннельный синдром и полиневропатия. Общее число выбранных сигналов для эксперимента было 24 000, и ЭМГ-сигналы пронумерованы соответственно как 0 – здоровый, 1 – карпальный туннельный синдром, 2 – кубитальный туннельный синдром, 3 – полиневропатия.

При программировании все данные разбиты на две части: первый из них составила 70 % от всей базы, и они выбраны как входные, 30 % от общей части выбраны как база тестирования. Ниже показан отрывок программы:

```
Epoch 22/30
16794/16794 [==============================] - 1s 65us/step - loss: 0.3960 - acc: 0.8011
Epoch 23/30
16794/16794 [==============================] - 1s 60us/step - loss: 0.3928 - acc: 0.8026
Epoch 24/30
16794/16794 [==============================] - 1s 61us/step - loss: 0.3954 - acc: 0.8018
Epoch 25/30
16794/16794 [==============================] - 1s 59us/step - loss: 0.3865 - acc: 0.8057
Epoch 26/30
16794/16794 [==============================] - 1s 61us/step - loss: 0.3892 - acc: 0.8020
Epoch 27/30
16794/16794 [==============================] - 1s 63us/step - loss: 0.3843 - acc: 0.8058
Epoch 28/30
16794/16794 [==============================] - 1s 64us/step - loss: 0.3896 - acc: 0.8024
Epoch 29/30
16794/16794 [==============================] - 1s 63us/step - loss: 0.3863 - acc: 0.8047
Epoch 30/30
16794/16794 [==============================] - 1s 63us/step - loss: 0.3892 - acc: 0.7992
7198/7198 [==============================] - 0s 55us/step

acc: 81.51%
```

Рис. 1. Точность нейронной сети после 30 эпох получена 81,51 %

Model Code:

```
model = Sequential()
model.add(Dense(8, input_dim = input_dim, activation = “relu”))
model.add(Dense(10, activation = “relu”))
model.add(Dense(10, activation = “relu”))
model.add(Dense(10, activation = “relu”))
model.add(Dense(4, activation = “softmax”))
model.compile(loss = “categorical_crossentropy”, optimizer = “adam”, metrics = [“accuracy”])
hist=model.fit(train_x, train_y, epochs = 30)
scores = model.evaluate(test_x, test_y)
print(«\n%s: %.2f%%» % (model.metrics_names[1], scores[1]*100))
```

Выбрано 30 эпох для данной сети, и результат показан на *рис. 1*.

Одним из широко используемых графиков для отладки нейронной сети является кривая потерь при обучении (*рис. 2*). Она дает снимок процесса обучения и направления, в котором сеть имеет тенденцию получать лучшие результаты и учиться. В течение эпохи функция потерь рассчитывается для каждого элемента данных. Построение кривой по эпохам дает потери для каждого подмножества всего набора данных. Еще одна кривая, широко используемая для понимания прогресса нейронных сетей, – это кривая точности. По мере уменьшения потерь при обучении производительность нейронной сети будет увеличиваться. Мы визуализировали эти две кривые вместе, чтобы понять влияние уменьшения потерь на повышение точности.

После обучения можно делать прогнозы с небольшой выборкой, удаленной из базы в начале. Для обучения нейронной сети необходимо было преобразовать виды в векторы. Значит, после предсказания необходимо провести обратную операцию, чтобы восстановить название ассоциированного вида. На *рис. 3* представлены результаты прогнозирования.

Можно видеть, что прогнозирование маленькой сети сначала дает хороший результат, так как система может четко прогнозировать входные сигналы. Только одна строка показывает неточное прогнозирование, и это связано с очень близкими значениями входных сигналов. Это можно устранить обновлением входной сети.

Заключение

Из *рис. 2* кривых точности и потерь при обучении можно видеть, что уменьшение потерь продолжается несмотря на то, что они пересекаются с кривой повышения точности после 5 эпох. Но после 25…30 эпох данные кривые стабилизируются, и это показывает хорошее число эпох. Обладая данной информацией, можно сократить время обучения, что важно не только для экономии времени, но также для экономии памяти машины. Используя этот метод, мы получили весьма хороший результат для классификации миографических заболеваний: так, из *рис. 3* видно, что результаты прогнозирования точны на 82 %.

Список литературы:

1. *Sangwoo N., Min K.S., Hyun A.K., Hyoun-Joong K., Il-Young J.* Development of Artificial Intelligence to Support Needle Electromyography Diagnostic Analysis // Healthcare Informatics Research. 2019. Vol. 25 (2). PP. 131-138.
2. *Park S.H., Lee S.P.* EMG pattern recognition based on artificial intelligence techniques // IEEE Transactions on Rehabilitation Engineering. 1998. Vol. 6 (4). PP. 400-405.
3. *Mahdi K., Mehran J.* Real-time intelligent pattern recognition algorithm for surface EMG signals // Biomedical Engineering Online. 2007. Vol. 6. PP. 1-24.

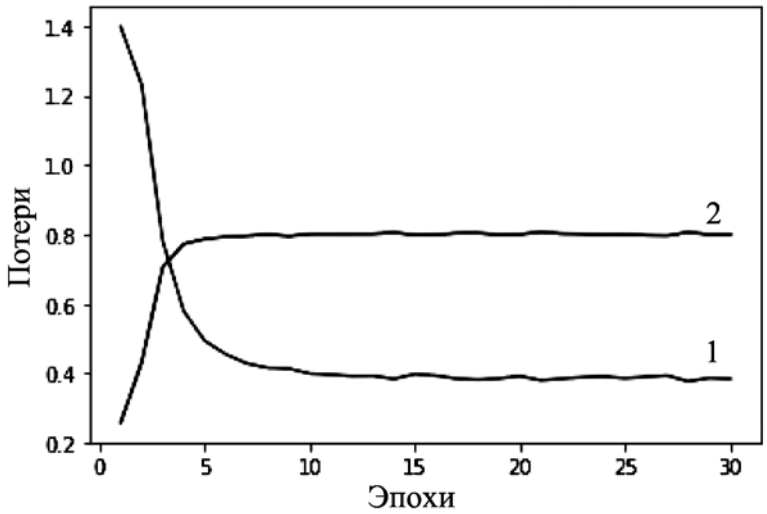

Рис. 2. Результат отношений кривой потерь при обучении и кривой точности: 1 – кривая потерь при обучении; 2 – кривая точности при обучении

```
the nn predict 1, and the species to find is 1
the nn predict 0, and the species to find is 0
the nn predict 3, and the species to find is 3
the nn predict 2, and the species to find is 2
the nn predict 0, and the species to find is 0
the nn predict 0, and the species to find is 0
the nn predict 1, and the species to find is 3
the nn predict 2, and the species to find is 2
```

Рис. 3. Результат прогнозирования сети

4. *Allard C.U., Campbell E., Phinyomark A., Laviolette F., Gosselin B., Scheme E.* Interpreting Deep Learning Features for Myoelectric Control: A Comparison With Handcrafted Features // Frontiers in Bioengineering and Biotechnology. 2020. Vol. 8. PP. 1-22.
5. *Panyawut S.I., Attawit C., Chatchai B., Songphon D., Ronachai P., Chusak T., Decho S.* Classification of 41 Hand and Wrist Movements via Surface Electromyogram Using Deep Neural Network // Frontiers in Bioengineering and Biotechnology. 2021. Vol. 9. PP. 1-15.
6. How to Use the Keras Functional API for Deep Learning / https://machinelearningmastery.com/keras-functional-api-deep-learning/ (дата доступа: 15.02.2022).
7. LabelEncoder (examples) / https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (дата доступа: 18.02.2022).
8. The Sequential model / https://keras.io/guides/sequential_model/ (дата доступа: 28.02.2022).
9. Keras – Models / https://www.tutorialspoint.com/keras/keras_models.htm (дата доступа: 28.02.2022).
10. Keras Model Sequential API VS Functional API / https://medium.com/analytics-vidhya/keras-model-sequential-api-vs-functional-api-fc1439a6fb10 (дата доступа: 05.03.2022).
11. Ways to build Keras Models / https://medium.com/nerd-for-tech/ways-to-build-keras-models-198f6a643944 (дата доступа: 05.03.2022).
12. Implicit and Explicit Input Layers in Keras Sequential Models / https://rukshanpramoditha.medium.com/implicit-and-explicit-input-layers-in-keras-sequential-models-733049f83a32 (дата доступа: 10.03.2022).
13. Multiclass EMG prediction with tensorflow keras / https://www.kaggle.com/aytacabdullayeva/multiclass-emg-prediction-with-tensorflow-keras (дата доступа: 14.03.2022).

Намик Таир оглы Абдуллаев,
д-р техн. наук, доцент,
кафедра «Биомедицинская техника»,
Азербайджанский технический университет,
Камаля Ширин гызы Пашаева,
канд. техн. наук, доцент,
кафедра автоматизации процессов,
Бакинская высшая школа нефти,
г. Баку, Азербайджан,
e-mail: nabdullayev.46@mail.ru

А.А. Кузнецов

Функции распределения R-R-интервалов ритмограмм

Аннотация

По обзору последних опубликованных работ автора представлен анализ форм функций распределений вероятностей случайных событий в виде R-R-интервалов. Обнаружены универсальность форм и полное соответствие нормальному закону распределения вероятностей. По трем выбранным ритмограммам (из 32-х) проведено исследование механизма формирования функции распределения по четырем моментам. Первым двум моментам соответствует асимптотическая форма приближения, а вторым двум – автоколебательная.

Введение

Начиная с 1997 года автором, совместно со студентами и аспирантами Владимирского государственного университета, проводились инициативные научно-исследовательские работы по общему направлению «Биофизика сердца» и акцентированно – ритмы сердца и их изменчивость. Работы были спровоцированы обращением двух реаниматологов областной клинической больницы с полученными ими результатами. В 1996 году были впервые опубликованы рекомендуемые стандарты к анализу ритмограмм [1]. Приблизительно в это же время были приоткрыты работы большой группы советских ученых в области космической медицины [2]. Они воспользовались «нехитрой математикой» и получили то, что получили. Почти сразу возникло недопонимание, которое преследует автора до сих пор со стороны медиков (особенно продвинутых), связанное с тем, что мышление физика никак не совпадает с мышлением медика. Здесь необходимо дать ответы на два вопроса: 1) медицина – наука или искусство; 2) возможно ли применять обозначенные методы исследования для анализа ритма сердца молодых здоровых людей (студентов 1-го, 2-го курсов). На первый вопрос ответ дал Петр Капица, четко разделивший ученого и технолога. Наука основывается на новом, неизведанном, которое необходимо открыть неизвестно где, кому и по какому направлению. Ученый должен много и достаточно поверхностно знать по самому широкому кругу вопросов. Логика у него смысловая, косвенная, иррациональная. Первичны интуиция и ощущение смысла. Технолог обязан много знать, значительно больше, чем ученый, но в узком направлении. Ведь он работает по принципу: «от известного»! Техногенная наука относится к материальной сфере. Технолог использует принцип сопоставления и аналогий, поэтому он инженер, изобретатель, но никак не ученый. Логика у него рациональная при причинно-следственном стиле мышления на основе первичности практики, опыта и инструментальных измерений. Искусство относится к нематериальной духовной сфере. Принятые трансцендентные отношения формируют «психосоциальные» восприятия. Автор назвал бы душу собственным ритмом человека, его основным тоном. С этим тоном ритм кровообращения обязан быть соизмерим [1]. В таком случае медицина находит свое место между искусством и наукой.

На второй вопрос ответ дал Р.М. Баевский в составе своей теории о донозологических состояниях человека [3]. Оба метода основаны на статистическом методе исследования. Но метод HRV имел вполне обозначенную область применения – ритмограммы людей с ишемической болезнью сердца (ИБС). Очень трудно представить стационарную ритмограмму человека с ИБС. Это возможно лишь в самом начале закупорки коронарных сосудов. Далее метод становится непригоден, так как для статистического метода основной и первой задачей является определение закона и функции распределения случайных событий [4], а для ритмограммы – определение значений R-R-интервалов. Для нестационарного сигнала распределение всегда многовершинное, поэтому смысл продолжения метода исчезает. В теории Баевского метод предназначен для исследования ритмов сердца не просто здоровых людей, а сверхздоровых, будущих космонавтов, при нагрузках физических, психических, соматических и эмоциональных за пределами возможного, вне рамок адаптации. Очевидно, ритмограмма таких людей будет стационарная при росте нагрузки. В этот